UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/622,022	07/17/2003	John B. Bley	WILY-01016US0	1684

28554 7590 07/03/2008
VIERRA MAGEN MARCUS & DENIRO LLP
575 MARKET STREET SUITE 2500
SAN FRANCISCO, CA 94105

EXAMINER	
CHOW, CHIH CHING	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
07/03/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/622,022	BLEY ET AL.	
	Examiner	Art Unit	
	CHIH-CHING CHOW	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *14 March 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-49* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-49* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *17 July 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to amendment dated March 14, 2008.

2. Per Applicants' request, claims 1, 29, 36 have been amended; claims 48 and 49 are new.

3. Claims 1-49 remain pending.

Response to Arguments

4. Applicants' arguments for Claims 1-49 have been fully considered respectfully by the examiner but they are not persuasive.

5. Applicants' arguments are basically in the following points:

- "Claim 1 is not anticipated by Chow because Chow does not disclose "adding new code to said existing object code including adding object code for an additional method that adds an additional function and object code that provides said result to said additional method, said object code that provides said result to said additional method is added to said first method."" (REMARKS dated 3/14/08, page 12)

Examiner's Response: In response to applicant's argument, see paragraph [0049] of current application, "FIG. 4 is a flowchart describing the process of modifying the existing object code in order to add new functionality that accesses return values and exceptions. In step 260, Probe Builder 4 receives the existing object code. In step 262, **Probe Builder 4 receives the new functionality, which can be new classes and methods that allow for monitoring the application**. In some embodiments, the new classes and methods can be in the form of a library. In step 264, the existing code is edited. In step 266, **all or part of the new functionality (e.g. the new classes/methods) is added to, combined with, or**

associated with the existing code. In step 268, the modified code (which includes the new functionality) is stored. In step 270, the modified code is run." – the new functionality is added to the existing code via Probe Builder, the specification doesn't indicate the new function is written in object code or 'and object code'; in fact, the object code is machine instructions, normally it's not able to be entered by a developer/programmer; the object code should be generated by the system or by the Probe Builder after the new function code is added, and eventually 'combined with, or associated with the existing code'. (see 35 USC § 112 rejection below**).**

- "However, this "new code" that is added to point to the label for the "additional method" does not also include "object code for [the] additional method." If the "additional method" is the exception handler, as the Examiner argues, the "additional method" is not part of the "new code" that is added to the "existing object code." Instead, the exception handler is part of the original code. Additionally, even assuming the "additional method" can be equated to the exception handler, this "additional method" does not add "an additional function" because the exception handler has the same "function" it did prior to the translation to high-level code. No new functionality is added by the "new code."…. Additionally, the "new code" that leads to the high-level code for the exception handler is not "object code." Instead, the "new code" is "high-level" code for the exception handler 'generated] to go to the level of the current [exception handler]" (REMARKS dated 3/14/08, pages 13-14)

Examiner's Response: In response to applicant's argument, the 'object code' is not included in the additional method, and the "new code" is not "object code" see argument response above, and USC § 112 rejection below. The Examiner didn't argue 'the "additional method" is not part of the "new code" that is added to the "existing object code." Instead, the exception handler is part of the original code.' – the examiner's citation in Office action dated 11/14/07, page 3, meant the new code has the same function as the exception handler, or adds new procedure to existing exception handler and produces results to existing code; therefore Chow's disclosure still teaches the subject claimed in the current invention.

- "Additionally, claim 9 is not anticipated by Chow because Chow does not disclose "adding start byte code; adjusting byte code indices; adding exit byte code."" (REMARKS dated 3/14/08, page 14)

Examiner's Response: Exception handler is as a block of code, which has a start point (entrance of the block) and an ending point (exit of the block), the code leads to the exception handler is the adjusting byte code indices (deviated from the normal execution of the program).

- "Claim 12 is not anticipated by Chow because Chow does not disclose "storing a result for a first method from an operand stack; preparing said operand stack for an innovation of a second method;…" "Chow does not disclose that this "result" is stored "from an operand stack." Nor does Chow disclose that the "operand stack" is prepared for "an invocation of a second method" or that the "result" is provide to the "second method" when the "second method" is invoked." (REMARKS dated 3/14/08, page 15)

Examiner's Response: See Chow's column 3, lines 21-27, "For example, computer system 21 **stores in its DASD 25 a bytecode program verifier 30 for verifying the id of a specified bytecode program**, and a bytecode program interpreter 31 for executing the specified bytecode program."; lines 40-49, "During normal execution of the bytecode program by bytecode program interpreter 31, **bytecode program interpreter 31 must continually monitor the operand stack** for overflows (i.e., attempting to add more data to the stack than the stack can store) and underflows (i.e., attempting to pop data off the stack when the stack is empty). Such stack monitoring must normally be performed for all bytecodes that change the status of the stack." – the result of the first method is stored in an operand stack.

6. Examiner is maintaining the 35 USC 112 and 102 Rejections. For the Applicants' convenience they are listed as following, with the amendments requested by the Applicants.

Claim Rejections - 35 USC § 112

7. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

8. Claim 1 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Claim 1 recites: "A method for adding functionality in order to access information, comprising: accessing existing object code, said existing object code includes a first method, said first method produces a result when said first method is executed; and adding new code to said existing

object code including adding object code for an ad additional method that adds an additional function and object code that provides said result to said additional method, said object code that provides said result to said additional method is added to said first method." -- see paragraph [0049] of current application, "FIG. 4 is a flowchart describing the process of modifying the existing object code in order to add new functionality that accesses return values and exceptions. In step 260, Probe Builder 4 receives the existing object code. In step 262, **Probe Builder 4 receives the new functionality, which can be new classes and methods that allow for monitoring the application**. In some embodiments, the new classes and methods can be in the form of a library. In step 264, the existing code is edited. In step 266, **all or part of the new functionality (e.g. the new classes/methods) is added to, combined with, or associated with the existing code**. In step 268, the modified code (which includes the new functionality) is stored. In step 270, the modified code is run." -- the new functionality is added to the existing code via Probe Builder, the specification doesn't indicate the new function is written in object code or 'and object code'; in fact, the object code is machine instructions, normally it's not able to be entered by a developer/programmer; the object code should be generated by the system or by the Probe Builder after the new function code is added, and eventually 'combined with, or associated with the existing code'. The examiner assumes the 'object code' is generated (not added directly) based on the new functionality that is added.

9. Claims 2-11, 48 and 49 depend on claim 1, they are rejected under 35 USC § 112 (2) for the same reason.

10. Claim 29 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter

which applicant regards as the invention. Claim 29 recites "One or more processor readable storage devices having processor readable code embodied on said processor readable storage devices, said processor readable code for programming one or more processors to perform a method comprising: accessing existing object code, said existing object code includes a first method, said first method produces a result when said first method is executed; and adding new code to said existing object code including adding object code for an additional method that adds an additional function and object code that provides said result to-said additional method, said object code that provides said result to said additional method is added to said first method" -- See paragraph [0049] of current application, "FIG. 4 is a flowchart describing the process of modifying the existing object code in order to add new functionality that accesses return values and exceptions. In step 260, Probe Builder 4 receives the existing object code. In step 262, **Probe Builder 4 receives the new functionality, which can be new classes and methods that allow for monitoring the application**. In some embodiments, the new classes and methods can be in the form of a library. In step 264, the existing code is edited. In step 266, **all or part of the new functionality (e.g. the new classes/methods) is added to, combined with, or associated with the existing code**. In step 268, the modified code (which includes the new functionality) is stored. In step 270, the modified code is run.– the new functionality is added to the existing code, it doesn't indicate that the object code is also added in the spec; in fact, the object code is machine instructions, normally it's not able to be added by developer/programmer; the object code should be generated by the system or the Probe Builder based on the added new function code. The examiner assumes the

'object code' is generated (not added directly) based on the new functionality that is added.

11. Claims 30-35 depend on claim 29, they are rejected under 35 USC § 112 (2) for the same reason.

12. Claim 36 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Claim 36 recites "An apparatus that adds functionality in order to access information, comprising: a communication interface; a processor readable storage device; and one or more processors in communication with said processor readable storage device and said communication interface, said one or more processors perform a method comprising: access existing object code, said existing object code includes a first method, said first method produces a result when said first method is executed, and adding new code to said existing object code including adding object code for an additional method that adds an additional function and object code that provides said result to said additional method, said object code that provides said result to said additional method is added to said first method." -- See paragraph [0049] of current application, "FIG. 4 is a flowchart describing the process of modifying the existing object code in order to add new functionality that accesses return values and exceptions. In step 260, Probe Builder 4 receives the existing object code. In step 262, **Probe Builder 4 receives the new functionality, which can be new classes and methods that allow for monitoring the application**. In some embodiments, the new classes and methods can be in the form of a library. In step 264, the existing code is edited. In step 266, **all or part of the new functionality (e.g. the new classes/methods) is added to, combined with, or**

associated with the existing code. In step 268, the modified code (which includes the new functionality) is stored. In step 270, the modified code is run.– the new functionality is added to the existing code, it doesn't indicate that the object code is also added in the spec; in fact, the object code is machine instructions, normally it's not able to be added by developer/programmer; the object code should be generated by the system or the Probe Builder based on the added new function code. The examiner assumes the 'object code' is generated (not added directly) based on the new functionality that is added.

13. Claims 37-40 depend on claim 36, they are rejected under 35 USC § 112 (2) for the same reason.

Claim Rejections - 35 USC § 102

14. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

15. Claims 1-6, 8-9, 12-18, 20-22, 25-27, 29-34, 36-39, 41-46 are rejected under 35 U.S.C. 102(b) as being anticipated by US Patent No. 6,131,187, by Chow et al., hereinafter "Chow".

As Per claim 1, Chow discloses:

- ***A method for adding functionality in order to access information, comprising: accessing existing object code, said existing object code includes a first method, said first method produces a result when said first method is executed; and adding new code to said ~~first method, said new code~~ existing object code including adding object code for an additional method that adds an additional function and object code that provides said***

result to said ~~an~~ additional method, said object code that provides said result to said additional method is added to said first method.

Chow's disclosure teaches a method to access existing object code, see Chow's Fig. 2, item 42, 'step through bytecode within a bytecode stream', and see Chow's column 3, line 65 into column 4, line 7, "An exception is a software error condition that **interrupts the flow of a bytecode program**. Within a **Java™** virtual machine, an exception always occurs when the **bytecode program executes** a **throw instruction** (*produces a result when said first method is executed).* The throw instruction passes control to an associated catch block. A catch block is a section of code designed to process the thrown exception. A catch block must immediately follow a **try block** or another **catch block**. The try block encloses a sequence of statements from which a throw can originate." – the original bytecode program is the first method. Further lines 40-67, "The **pertinent information** preferably includes a program counter, a stack pointer, an exception handler program counter (if present), the catch type for the exception handler (in necessary), other information such as branch address, and various flags. All pertinent information in each entry 36 of bytecode information array 35 is obtained by **stepping through each bytecode** within a given bytecode stream (executing the first method), as depicted in block 42." And the pseudo code **Procedure StepThruCode** (access existing object code). Further, see pseudo code in column 5, lines 10-35, wherein the code (try and catch) lead to the exception handler is the new code, the new code provides results (the exception) to the additional method (exception handler), which is the procedures within the exception handler; the

additional method is for tracing and monitoring purposes, such as building up the bytecode information array (*adding new code to said existing object code, said new code provides said result to an additional method).* As to the 'adds an additional function and object code' feature see **35 USC § 112 (2)** rejection above.

As Per claim 2, Chow discloses:

- ***The method according to claim 1, wherein:***

Said result includes a data item to be returned by said first method.

Claim 1 rejection is incorporated, further see Chow's column 4, lines 40-66, the 'pc', 'sp', 'eh', …etc. are all data items returned by executing the procedure (the first method).

As Per claim 3, Chow discloses:

- ***A method according to claim 1, wherein: said result includes a reference to an exception.***

Claim 1 rejection is incorporated, further see Chow's column 4, lines 2-4, "the throw instruction passes control to an associated catch block. A catch block is a section of code designed to **process the thrown exception**." – the result includes a reference to an exception.

As Per claim 4, Chow discloses:

- ***A method according to claim 1, wherein said step of adding new code includes: adding code that stores said result for said first method from an operand stack; adding code that prepares said operand stack for an***

invocation of said additional method; adding code that invokes said additional method, including providing said result to said additional method; and adding code that resets said operand stack with respect to said result to a state existing prior to storing said result.

Claim 1 rejection is incorporated, further see Chow's column 3, lines 42-46, "During **normal execution of the bytecode program** by bytecode program interpreter 31, bytecode program interpreter 31 must continually **monitor the operand stack** for overflows (i.e., attempting to **add more data to the stack** than the stack can store) and underflows (i.e., attempting to pop data off the stack when the stack is empty)." -- *operand stack for an invocation.* Also see Chow's column 5, lines 63-66, "If no exception occurred for an instruction working on the same operand(s), then the **generation of exception handling code**, such as Null PointerException and ArrayOutOfBoundException, for all the subsequent instructions of the same opcode and operands is not required." -- *adding code that resets said operand stack.*

As Per claim 5, Chow discloses:

- ***A method according to claim 4, wherein said step of adding new code further includes: adding code that returns said result after resetting said operand stack, said result is a return value.***

See claim 4 rejection, further see Chow's column 3, lines 29-36, "Bytecode program verifier 30 is an executable program that verifies operand data type compatibility and **stack manipulations** properness in a specified bytecode program prior to the execution of the bytecode program by processor 22

under bytecode program interpreter 31. Each bytecode program has an associated **verification status value that is initially set** when the bytecode program is being downloaded from another location, such as a file server." *(reset operand stack).*

As Per claim 6, Chow discloses:

- ***A method according to claim 4, wherein: said result includes an exception; and said step of adding new code further includes adding code that throws said exception after said step of resetting, said result represents an exception.***

Claim 4 rejection is incorporated, for rest of claim 6 feature see claim 1 rejection.

As Per claim 8, Chow discloses:

- ***A method according to claim 1, wherein: said first method is a JAVA method.***

Claim 4 rejection is incorporated, for JAVA feature see Chow's column 3, line 65 into column 4, line 7, "An exception is a software error condition that **interrupts the flow of a bytecode program**. Within a **Java™** virtual machine, an exception always occurs when the **bytecode program executes** a **throw instruction** (*produces a result when said first method is executed).* The throw instruction passes control to an associated catch block. A catch block is a section of code designed to process the thrown exception", and column 5, lines 35-39, "As has been described, the present invention provides an improved method for translating exception handling semantics

of a bytecode class file within a computer system. Instead of following the exception **handling mechanism of a Java™ class file**, the present invention analyzes the **exception handling frames** defined in the class file."

As Per claim 9, Chow discloses:

- ***A method according to claim 1, wherein said step of adding new code includes: adding start byte code; adjusting byte code indices; adding exit byte code; and modifying an exception table for said first method.***

Claim 1 rejection is incorporated, further for of claim 9 feature see Chow's column 1, lines 52-61, "The exception handling semantics, embedded within an exception handling structure (or exception framelist) of a bytecode class file, are designed to allow the bytecode interpreter to **handle exceptions that occur** during an enclosed try statement. However, the exception handling structure prevents the translated high-level code from being optimized by the compiler because the artificial exception ranges **kept in the exception table** of a method preclude the final natively compiled instructions from being rescheduled for optimal performance." And Chow's column 4, Table I contains information similar to the start byte code, adjusting byte code, indices, counts, for exception handlers.

As Per claim 12, Chow discloses:

- ***A method to provide access information, comprising: storing a result for a first method from an operand stack; preparing said operand stack for an invocation of a second method; invoking said second method, including providing said result to said second method; and resetting said operand***

stack with respect to said result to a state existing prior to said step of storing said result.

Chow's disclosure is for accessing stored information see Chow's Fig. 1, and description in column 3, lines 42-47, "bytecode program interpreter 31 must continually monitor the **operand stack** for overflows (i.e., attempting to **add more data to the stack than the stack can store**) and underflows (i.e., attempting to pop data off the stack when the stack is empty). Such stack monitoring must normally be performed for all bytecodes that change the status of the stack." Also see Chow's column 3, lines 21-27, "For example, computer system 21 **stores in its DASD 25 a bytecode program verifier 30 for verifying the id of a specified bytecode program**, and a bytecode program interpreter 31 for executing the specified bytecode program."; lines 40-49, "During normal execution of the bytecode program by bytecode program interpreter 31, **bytecode program interpreter 31 must continually monitor the operand stack** for overflows (i.e., attempting to add more data to the stack than the stack can store) and underflows (i.e., attempting to pop data off the stack when the stack is empty). Such stack monitoring must normally be performed for all bytecodes that change the status of the stack." – the result of the first method is stored in an operand stack.

As Per claim 13, Chow discloses:

- ***A method according to claim 12, wherein: said result includes a data item to be returned by said first method.***

Claim 12 rejection is incorporated, further see claim 2 rejection.

As Per claim 14, Chow discloses:

- ***A method according to claim 13, further comprising: returning said result after said step of resetting.***

Claim 13 rejection is incorporated, further see claim 5 rejection.

As Per claim 15, Chow discloses:

- ***A method according to claim 12, wherein: said result includes a reference to an exception.***

Claim 12 rejection is incorporated, further see claim 3 rejection.

As Per claim 16, Cow discloses:

- ***A method according to claim 15, further comprising: throwing said exception after said step of resetting.***

For claim 15 feature see claim 15 rejection, for rest of claim 16 feature see claim 5 rejection.

As Per claim 17, Chow discloses:

- ***A method according to claim 12, further comprising: performing said second method in response to said step of invoking.***

For claim 12 feature see claim 12 rejection, for rest of claim 17 feature see claim 7 rejection.

As Per claim 18, Chow discloses:

- ***A method according to claim 12, further comprising: performing one or more instructions of said first method prior to said step of storing said result, said step of performing one or more instructions includes generating said result.***

For claim 12 feature see claim 12 rejection, for rest of claim 18 feature see claim 1 rejection.

As Per claim 20, Chow discloses:

- ***A method according to claim 12, wherein: said first method is a JAVA method.***

For claim 12 feature see claim 12 rejection, for rest of claim 20 feature see claim 8 rejection.

As Per claim 21, Chow discloses:

- ***A method according to claim 12, further comprising: modifying byte code for said first method to add new code that performs said steps of storing, preparing, invoking and resetting.***

For claim 12 feature see claim 12 rejection, for rest of claim 21 feature see claim 9 rejection.

As Per claim 22, Chow discloses:

- ***A method according to claim 21, wherein said step of modifying includes: adding start byte code; adjusting byte code indices; adding exit byte code; and modifying an exception table for said first method.***

For claim 21 feature see claim 21 rejection, for rest of claim 22 feature see

claim 9 rejection.

As Per claim 25, Chow discloses:

- ***A method according to claim 12, further comprising: performing said second method, including accessing said result.***

Claim 12 rejection is incorporated, further see claims 1, 4 rejections.

As Per claim 26, Chow discloses:

- ***A method according to claim 12, further comprising: performing said second method, including storing said result for use outside of a thread that includes said first method.***

Claim 12 rejection is incorporated, further see claim 1 rejection.

As Per claim 27, Chow discloses:

- ***A method according to claim 12, further comprising: performing said second method in response to said step of invoking, said second method stores said result; performing one or more instructions of said first method prior to said step of storing said result, said step of performing one or more instructions includes generating said result, said first method is a JAVA method; and modifying byte code for said first method to add new code that performs said steps of storing, preparing, invoking and resetting.***

For claim 12 feature see claim 12 rejection, for rest of claim 27 feature see claims 1, and 8 rejections.

As Per claim 29, Chow discloses:

- ***One or more processor readable storage devices having processor readable code embodied on said processor readable storage devices, said processor readable code for programming one or more processors to perform a method comprising: accessing existing object code, said existing object code includes a first method, said first method produces a result when said first method is executed; and adding new code to said ~~first method, said new code~~ existing object code including adding object code for an additional method that adds an additional function and object code that provides said result to ~~an~~ said additional method, said object code that provides said result to said additional method is added to said first method.***

Chow's teaching also applies for one or more processor readable storage devices having processor readable code; claim 29 is one or more processor readable storage devices' version of claim 1, it is rejected on the same basis as claim 1. As to the 'adding object code' feature see **35 USC § 112 (2)** rejection above.

As Per claim 30, Chow discloses:

- ***One or more processor readable storage devices according to claim 29, wherein: said result is a data item to be returned by said first method.***

Claim 29 rejection is incorporated, further see claim 2 rejection.

As Per claim 31, Chow discloses:

- ***One or more processor readable storage devices according to claim 29, wherein: said result is a reference to an exception.***

Claim 29 rejection is incorporated, further see claim 3 rejection.

As Per claim 32, Chow discloses:

- ***One or more processor readable storage devices according to claim 29, wherein said step of adding new code includes: adding code that stores said result for said first method from an operand stack; adding code that prepares said operand stack for an invocation of said additional method; adding code that invokes said additional method, including providing said result to said additional method; and adding code that resets said operand stack with respect to said result to a state existing prior to storing said result.***

For claim 29 feature see claim 29 rejection, for rest of claim 32 feature see claim 4 rejection.

As Per claim 33, Chow discloses:

- ***One or more processor readable storage devices according to claim 29, wherein: said first method is JAVA method.***

For claim 29 feature see claim 29 rejection, for rest of claim 33 feature see claim 8 rejection.

As Per claim 34, Chow discloses:

- ***One or more processor readable storage devices according to claim 29, wherein said step of adding new code includes: adding start byte code; adjusting byte code indices; adding exit byte code; and modifying an exception table for said first-method.***

Claim 29 rejection is incorporated, further see claim 9 rejection.

As Per claim 36, Chow discloses:

- ***An apparatus that adds functionality in order to access information, comprising: a communication interface; a processor readable storage device; and one or more processors in communication with said processor readable storage device and said communication interface, said one or more processors perform a method comprising: access existing object code, said existing object code includes a first method, said first method produces a result when said first method is executed, and adding new code to said ~~first method, said new code~~ existing object code including adding object code for an additional method that adds an additional function and object code that provides said result to ~~an~~ said additional method, said object code that provides said result to said additional method is added to said first method.***

Chow's teaching also applies for an apparatus, which comprising a communication interface (see Chow's Fig. 1) and a processor readable storage device (see Chow's Fig. 1), and one or more processors in communication with the communication interface; claim 36 is an apparatus version of claim 1, it is rejected on the same basis as claim 1. As to the 'adding object code' feature see **35 USC § 112 (2)** rejection above.

As Per claim 37, Chow discloses:

- ***An apparatus according to claim 36, wherein: said result is a data item or a reference to an exception.***

Claim 36 rejection is incorporated, further see claim 3 rejection.

As Per claim 38, Chow discloses:

- ***An apparatus according to claim 36, wherein said step of adding new code includes: adding code that stores said result for said first method from an operand stack; adding code that prepares said operand stack for an invocation of said additional method; adding code that invokes said additional method, including providing said result to said additional method; and adding code that resets said operand stack with respect to said result to a state existing prior to storing said result.***

Claim 36 rejection is incorporated, further see claim 4 rejection.

As Per claim 39, Nilsson discloses:

- ***An apparatus according to claim 36, wherein said step of adding new code includes: adding start JAVA byte code; adjusting JAVA byte code indices; adding exit JAVA byte code; and modifying an exception table for said first method.***

For claim 36 feature see claim 36 rejection, for rest of claim 39 feature see claim 8 and claim 9 rejection.

As Per claim 41, Chow discloses:

- ***An apparatus that adds functionality to existing code in order to access information, comprising: a communication interface; a processor readable storage device; and one or more processors in communication with said processor readable storage device and said communication interface, said***

one or more processors perform a method comprising: storing a result for a first method from an operand stack, preparing said operand stack for an invocation of a second method, invoking said second method, including providing said result to said second method, and resetting said operand stack with respect to said result to a state existing prior to said step of storing said result.

Claim 41 is an apparatus version of claim 1, it is rejected on the same basis as claims, 1, 4, 36 rejections.

As Per claim 42, Chow discloses:

- ***An apparatus according to claim 41, wherein: said result is a data item to be returned by said first method.***

Claim 41 rejection is incorporated, further see claim 2 rejection.

As Per claim 43, Chow discloses:

- ***An apparatus according to claim 41, wherein: said result is a reference to an exception.***

Claim 41 rejection is incorporated, further see claim 3 rejection.

As Per claim 44, Nilsson discloses:

- ***An apparatus according to claim 41, wherein: said first method is a JAVA method.***

For claim 41 feature see claim 41 rejection, for rest of claim 44 feature see claim 8 rejection.

As Per claim 45, Chow discloses:

- ***An apparatus according to claim 41, wherein said method further comprises: modifying byte code for said first method to add new code that performs said steps of storing, preparing, invoking and resetting.***

Claim 41 rejection is incorporated, further see claim 9 rejection.

As Per claim 46, Chow discloses:

- ***An apparatus according to claim 45, wherein said step of modifying includes the steps of: adding start byte code; adjusting byte code indices; adding exit byte code; and modifying an exception table for said first method.***

Claim 45 rejection is incorporated, further see claim 9 rejection.

Claim Rejections - 35 USC § 103

16. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

17. Claims 7, 10, 11, 19, 23, 24, 28, 35, 40, 47, and 48, are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,131,187 by Chow et al., hereinafter "Chow", in view of US Patent No. 5, 628,016 by Kukol, hereinafter "kukol".

As Per claim 7,

- ***A method according to claim 4, wherein said step of adding new code further includes: adding code that jumps to a subroutine representing a Finally block after invoking said additional method; and adding code that is to be executed after returning from said subroutine.***

Claim 4 rejection is incorporated, Chow teaches all aspects of claim 7, but he does not disclose 'Finally block' explicitly, however, Kukol teaches in an analogous prior art, see Kukol's Fig. 9, which illustrating occurrence of a **try/finally block** within a setjmp block."

It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Chow's disclosure of the exception handler, with try and catch blocks by the 'finally block' taught by Kukol. The modification would be obvious because one of ordinary skill in the art would be motivated by the finalization code is executed (See Kukol's column 24, lines 50-58).

As Per claim 10,

- ***A method according to claim 9, wherein said step of adding exit byte code includes: adding byte code to report said result and jump to a subroutine representing a Finally block; adding byte code to report an exception and jump to said subroutine representing said Finally block; and adding byte code for said subroutine representing said Finally block.***

Claim 9 rejection is incorporated, Chow teaches all aspects of claim 10, but he does not disclose 'Finally block' explicitly, however, Kukol teaches in an analogous prior art, see Kukol's Fig. 9, which illustrating occurrence of a **try/finally block** within a setjmp block."

It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Chow's disclosure of the exception handler, with try and catch blocks by the 'finally block' taught by Kukol. The modification would be obvious because one of ordinary skill in the art would be motivated by the finalization code is executed (See Kukol's column 24, lines 50-58).

As Per claim 11,

- ***A method according to claim 1, wherein said step of adding new code includes: adding Try-Finally functionality.***

Claim 1 rejection is incorporated, Chow teaches all aspects of claim 10, but he does not disclose 'Finally block' explicitly, however, Kukol teaches in an analogous prior art, see Kukol's Fig. 9, which illustrating occurrence of a **try/finally block** within a setjmp block."

It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Chow's disclosure of the exception handler, with try and catch blocks by the 'finally block' taught by Kukol. The modification would be obvious because one of ordinary skill in the art would be motivated by the finalization code is executed (See Kukol's column 24, lines 50-58).

As Per claim 19,

- ***A method according to claim 12, further comprising: returning said result subsequent to said step of resetting; jumping to a subroutine representing a Finally block after invoking said second method and prior***

to returning said result; and returning from said subroutine prior to returning said result.

For claim 12 feature see claim 12 rejection, for rest of claim 19 feature see claim 7 rejection.

As Per claim 23,

- ***A method according to claim 22, wherein said step of adding exit bytes code includes: adding byte code to report said result and jump to a subroutine representing a Finally block; adding byte code to report an, exception and jump to said subroutine representing said Finally block; and adding byte code for said subroutine representing said Finally block.***

Claim 22 rejection is incorporated, further for see claim 10 rejection.

As Per claim 24,

- ***A method according to claim 21, wherein said step of modifying includes: adding Try-Finally functionality.***

Claim 21 rejection is incorporated, further for see claim 11 rejection.

As Per claim 28,

- ***A method according to claim 27, further comprising: returning said result; jumping to a subroutine representing a Finally block after invoking said second method and prior to returning said result; and returning from said subroutine prior to returning said result.***

For claim 27 feature see claim 27 rejection, for rest of claim 28 feature see claim 10 rejection.

As Per claim 35,

- ***One or more processor readable storage devices according to claim 34, wherein said step of adding exit byte code includes: adding byte code to report said result and jump to a subroutine representing a Finally block; adding byte code to report an exception and jump to said subroutine representing said Finally block; and adding byte code for said subroutine representing said Finally block.***

Claim 34 rejection is incorporated, further see claim 10 rejection.

As Per claim 40,

- ***An apparatus according to claim 39, wherein said step of adding exit byte code includes: adding byte code to report said result and jump to a subroutine representing a Finally block; adding byte code to report an exception and jump to said subroutine representing said Finally block; and adding byte code for said subroutine representing said Finally block.***

For claim 39 feature see claim 39 rejection, for rest of claim 40 feature see claim 7 rejection.

As Per claim 47,

- ***An apparatus according to claim 46, wherein said step of adding exit byte code includes: adding byte code to report said result and jump to a subroutine representing a Finally block; adding byte code to report an exception and jump to said subroutine representing said Finally block; and adding byte code for said subroutine representing said Finally block.***

Claim 46 rejection is incorporated, further see claim 10 rejection.

As Per claim 48,

- ***A method according to claim 1, wherein:***

said step of adding object code for said additional method that adds said additional function includes adding a tracer for said first method.

Claim 1 rejection is incorporated, Kukol teaches 'tracer' see Kukol's column 21 last two lines into column 22 first 5 lines, " The context table is used to **keep track of which part of a given function code is executing** *(tracer)*. When a function body is executing and an exception occurs at some random point, the context table allows the system to **determine what part of the function was executing**, what cleanup is necessary before the stack frame may be thrown away, and whether there exists any try blocks that are active."

18. Claim 49 is rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,131,187 by Chow et al., hereinafter "Chow", in view of a prior art of record, US Patent No. 6,934,832, by Van Dyke et al., hereinafter "Van Dyke".

As Per claim 49,

- ***A method according to claim 1, wherein:***

said step of adding object code for said additional method that adds said additional function includes adding a timer for said first method.

Claim 1 rejection is incorporated, Chow teaches all aspects of claim 49, but he does not disclose 'a timer' explicitly, however, Van Dyke teaches in an analogous prior art, see Van Dyke column 85, lines 13-16, "n the embodiment discussed at length in the following sections, TAXi divides the possible event space by space (pages), **time (using the Probe timer)**, and event code (the same event code 402 used in profiling)."

It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Chow's disclosure of the exception handler, with try and catch blocks by the 'finally block' taught by Van Dyke. The modification would be obvious because one of ordinary skill in the art would understand the time is an important factor of effective probing (See Van Dyke's column 86, lines 10-25).

Conclusion

19. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Reese et al., US Patent No. 6,941,545, discloses an instruction pipeline and memory access unit execute instructions in a logical address space of a memory of the computer. An address translation circuit translates address references generated by the program from the program's logical address space to the computer's physical address space. A mode of execution of the instructions may be changed without further intervention when execution flows from the first region to the second, or the mode may be changed by an exception handler when the computer takes an exception when execution flows from the first region to the second.

Kramskoy et al., US 2002/0112227, discloses a dynamic compiler and method of compiling code to generate a dominate path and handle exceptions. The dynamic compiler includes an execution history recorder that is configured to record the number of times a fragment of code is interpreted.

20. The following summarizes the status of the claims:

35 USC § 112 (2) rejection: Claims 1-11, 48 and 49; 29-40

35 USC § 102 rejection: Claims 1-6, 8-9, 12-18, 20-22, 25-27, 29-34, 36-39, 41-46

35 USC § 103 rejection: Claims 7, 10, 11, 19, 23, 24, 28, 35, 40, 47, 48, 49

Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 571-272-3693. The examiner can normally be reached on 7:30am - 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300. Any inquiry of a general nature of relating to the status of this application should be directed to the **TC2100 Group receptionist: 571-272-2100**.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Chih-Ching Chow/
Examiner, Art Unit 2191
6/25/2008

/Ted T. Vo/
Primary Examiner, Art Unit 2191